Kalman Filter

Raul Queiroz Feitosa

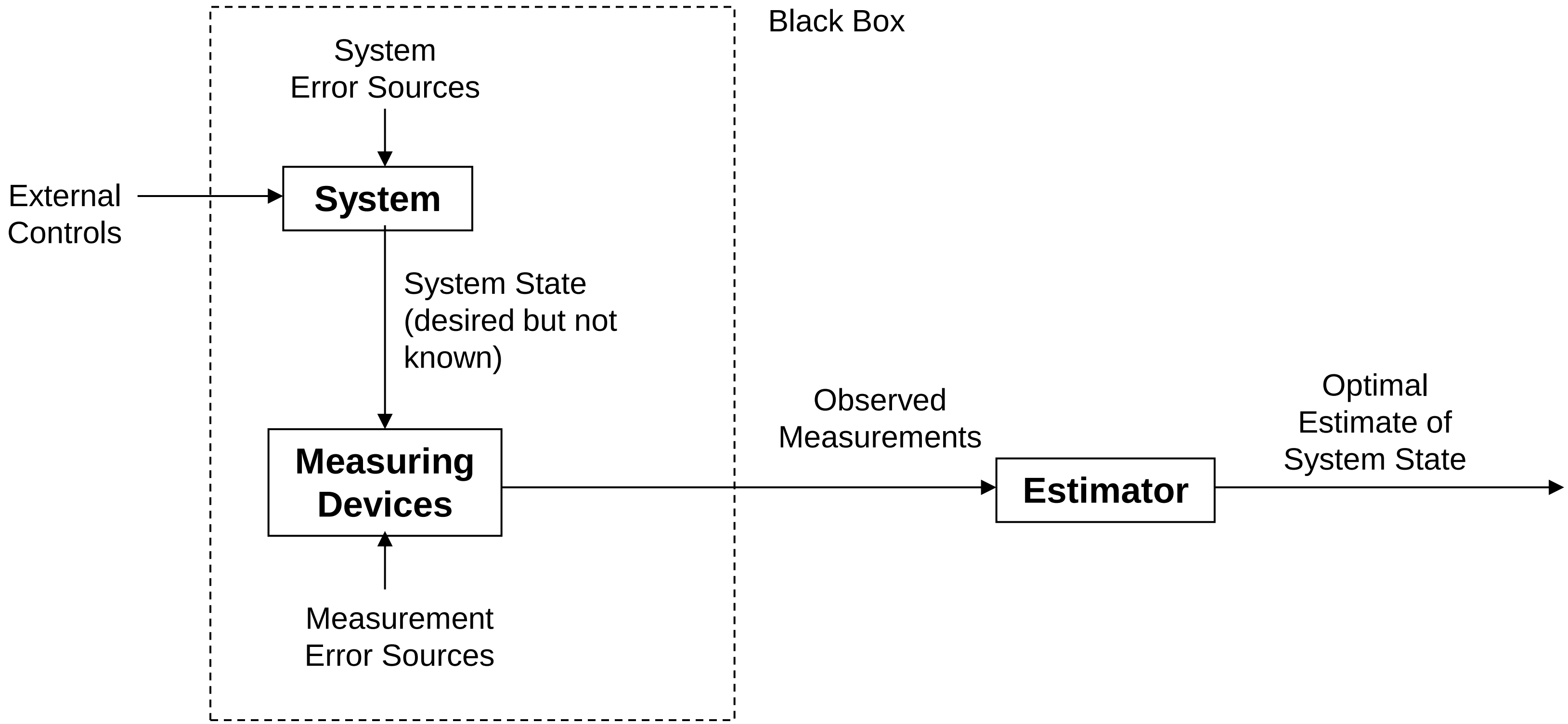
Rudolf Emil Kálmán

- Born in 1930 in Hungary
- BS and MS from MIT
- PhD 1957 from Columbia
- Filter developed in 1960-61
- IEEE Medal of Honor in 1974
- National Medal of Science 2009
- Retired.

Applications

- Aerospace,
- Marine navigation,
- Nuclear power plant instrumentation,
- Demographic modeling,
- Manufacturing,
- Tracking,
- …

The Problem



- System state cannot be measured directly
- Need to estimate "optimally" from measurements

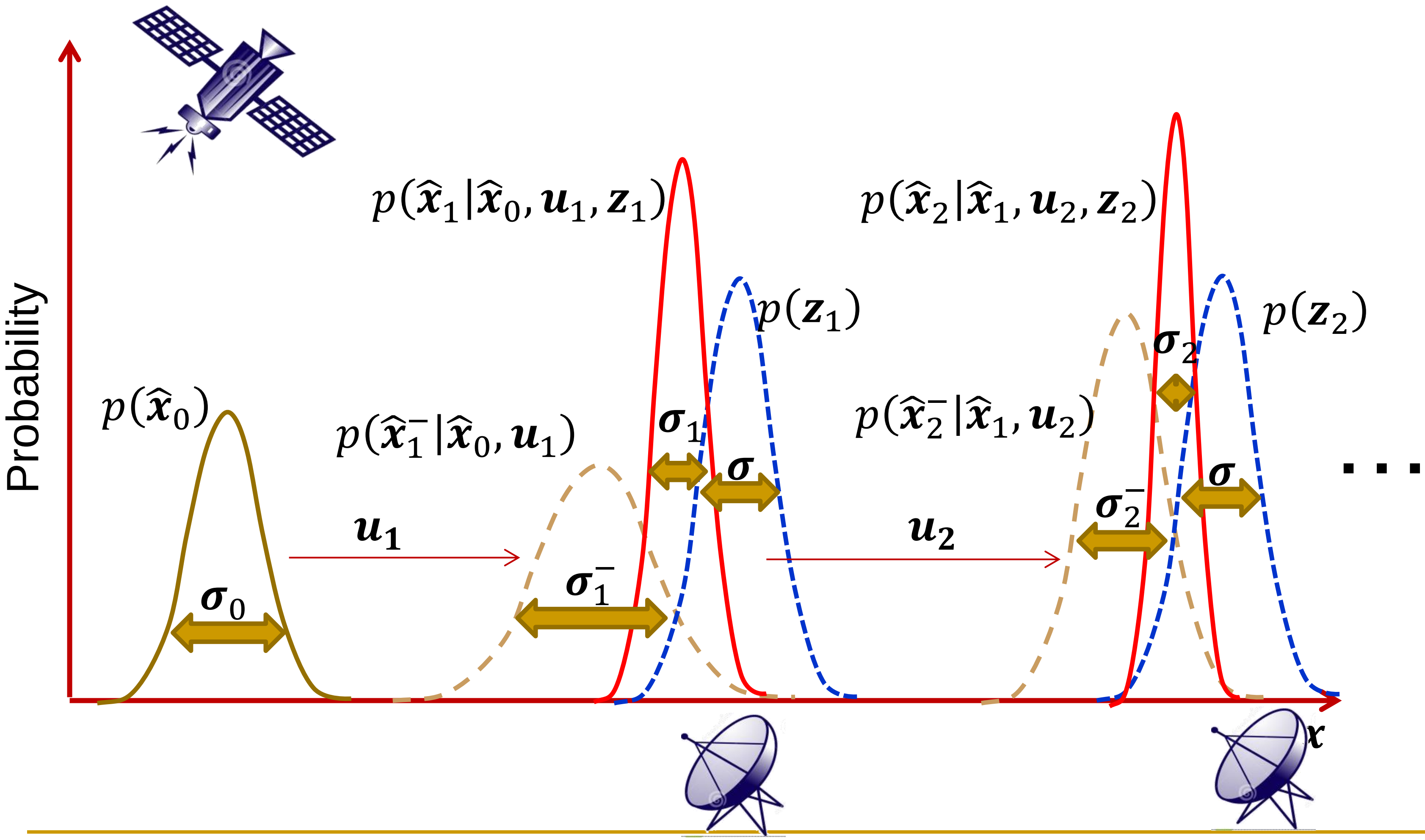
What is a Kalman Filter?

- **Recursive** data processing algorithm
  - Doesn't need to store all previous measurements and reprocess all data each time step
- Generates **optimal** estimate of desired quantities given the set of measurements
- Optimal?
  - For linear system and white Gaussian errors, Kalman filter delivers "best" estimate based on all previous measurements.
  - For non-linear system optimality is not guaranteed, but it works fine.

Conceptual Overview

- Simple example to motivate the workings of the Kalman Filter
- Theoretical Justification to come later – for now just focus on the concept
- Important: Prediction and Correction

Conceptual Overview



Conceptual Overview

Lessons so far

Make prediction based on previous data - $\hat{\boldsymbol{x}}_k^-$, $\boldsymbol{\sigma}_k^-$

Take measurement – $\boldsymbol{z}_k$, $\boldsymbol{\sigma}$

Optimal estimate ($\hat{\boldsymbol{x}}_k$) = Prediction ($\hat{\boldsymbol{x}}_k^-$) + …
(Kalman Gain) * (Measurement ($\boldsymbol{z}_k$) – Prediction ($\hat{\boldsymbol{x}}_k^-$))

Variance of estimate ($\boldsymbol{\sigma}_k$) = Variance of prediction ($\boldsymbol{\sigma}_k^-$) * (1 – Kalman Gain)

The state equation

state matrix at time k

state matrix at time $k-1$

control input at time k

$$\boldsymbol{x}_k = \boldsymbol{A}\boldsymbol{x}_{k-1} + \boldsymbol{B}\boldsymbol{u}_k + \boldsymbol{w}_{k-1}$$

relates $\boldsymbol{x}_{k-1}$ to $\boldsymbol{x}_k$, in the absence of either a driving function or process noise. It might change with k.

relates the <u>optional</u> control input $\boldsymbol{u}_{k-1}$ to the state $\boldsymbol{x}_k$. It might change with k.

process noise with distribution $\boldsymbol{p}(\boldsymbol{w}) = \mathcal{N}(\boldsymbol{0}, \boldsymbol{Q})$

process noise covariance

The measurement equation

state matrix at time k

measurement at time k

$$\boldsymbol{z}_k = \boldsymbol{H}\boldsymbol{x}_k + \boldsymbol{v}_k$$

measurement noise with distribution $\boldsymbol{p}(\boldsymbol{v}) = \mathcal{N}(\boldsymbol{0}, \boldsymbol{R})$

relates $\boldsymbol{z}_k$ to $\boldsymbol{x}_k$, in the absence of measurement noise. It might change with k.

measurement noise covariance

The errors

state matrix

a priori state estimate based on process history

a priori error estimate

$$e_k^- = x_k - \hat{x}_k^-$$

a posteriori state estimate based on process history AND on measurement.

a posteriori error estimate

$$e_k = x_k - \hat{x}_k$$

a priori covariance error estimate

$$P_k^- = E\left[e_k^- e_k^{-T} \right]$$

a posteriori covariance error estimate

$$P_k = E\left[e_k e_k^T \right]$$

Kalman Filter task: $\hat{x}_k^- \rightarrow \hat{x}_k$ **AND** $P_k^- \rightarrow P_k$

Time update (predictor)

1. Update expected value for the state matrix

$$\hat{\boldsymbol{x}}_k^- = \boldsymbol{A}\hat{\boldsymbol{x}}_{k-1} + \boldsymbol{B}\boldsymbol{u}_k$$

control input at time k

a priori **state estimate at time** k

a posteriori **state estimate at time** $k-1$**.**

2. Update error covariance matrix **P**

a priori covariance **error estimate at time** k

$$\boldsymbol{P}_k^- = \boldsymbol{A}\boldsymbol{P}_{k-1}\boldsymbol{A}^T + \boldsymbol{Q}$$

process noise covariance

a posteriori covariance **error estimate at time** $k-1$

Measurement update(corrector)

1. Update expected value

Kalman gain at time k

measurement at time k

$$\hat{\boldsymbol{x}}_k = \hat{\boldsymbol{x}}_k^- + \boxed{\boldsymbol{K}_k(\boldsymbol{z}_k - \boldsymbol{H}\hat{\boldsymbol{x}}_k^-)}$$

a posteriori **state estimate at time** k.

a priori **state estimate at time** k.

2. Update error covariance matrix

***a posteriori* covariance error estimate**

$$\boldsymbol{P}_k = (I - \boldsymbol{K}_k\boldsymbol{H})\boldsymbol{P}_k^-$$

***a priori* covariance error estimate**

The Kalman Gain

The optimal Kalman gain $\boldsymbol{K}_k$ is

measurement
noise
covariance

$$\boldsymbol{K}_k = \boldsymbol{P}_k^- \boldsymbol{H}^T (\boldsymbol{H}\boldsymbol{P}_k^- \boldsymbol{H}^T + \boldsymbol{R})^{-1}$$

corresponds to the value that minimizes the trace of $\boldsymbol{P}_k$, the covariance of posterior state estimate (see Brown & Hwang 2012 ch. 4 for details)

Summary

Process to be estimated

$$\boldsymbol{x}_k = \boldsymbol{A}\boldsymbol{x}_{k-1} + \boldsymbol{B}\boldsymbol{u}_k + \boldsymbol{w}_{k-1}$$ Process Noise ($\boldsymbol{w}$) with covariance $\boldsymbol{Q}$

$$\boldsymbol{z}_k = \boldsymbol{H}\boldsymbol{x}_k + \boldsymbol{v}_k$$ Measurement Noise ($\boldsymbol{v}$) with covariance $\boldsymbol{R}$

Kalman Filter

Predicted: $\hat{\boldsymbol{x}}_k^-$ is estimated based on measurements at previous time-steps

$$\hat{\boldsymbol{x}}_k^- = \boldsymbol{A}\hat{\boldsymbol{x}}_{k-1} + \boldsymbol{B}\boldsymbol{u}_k$$

$$\boldsymbol{P}_k^- = \boldsymbol{A}\boldsymbol{P}_{k-1}\boldsymbol{A}^T + \boldsymbol{Q}$$

Corrected: $\hat{\boldsymbol{x}}_k$ has additional information – the measurement at time k

$$\left.\begin{aligned}\hat{\boldsymbol{x}}_k &= \hat{\boldsymbol{x}}_k^- + \boldsymbol{K}_k(\boldsymbol{z}_k - \boldsymbol{H}\hat{\boldsymbol{x}}_k^-)\\ \boldsymbol{P}_k &= (I - \boldsymbol{K}_k\boldsymbol{H})\boldsymbol{P}_k^-\end{aligned}\right\} \quad \boldsymbol{K}_k = \boldsymbol{P}_k^-\boldsymbol{H}^T(\boldsymbol{H}\boldsymbol{P}_k^-\boldsymbol{H}^T + \boldsymbol{R})^{-1}$$

Recursive data processing

Prediction

(1) $\hat{\boldsymbol{x}}_k^- = \boldsymbol{A}\hat{\boldsymbol{x}}_{k-1} + \boldsymbol{B}\boldsymbol{u}_k$

(2) $\boldsymbol{P}_k^- = \boldsymbol{A}\boldsymbol{P}_{k-1}\boldsymbol{A}^T + \boldsymbol{Q}$

Time update

Correction

(1) $\boldsymbol{K}_k = \boldsymbol{P}_k^-\boldsymbol{H}^T(\boldsymbol{H}\boldsymbol{P}_k^-\boldsymbol{H}^T + \boldsymbol{R})^{-1}$

(2) $\hat{\boldsymbol{x}}_k = \hat{\boldsymbol{x}}_k^- + \boldsymbol{K}_k(\boldsymbol{z}_k - \boldsymbol{H}\hat{\boldsymbol{x}}_k^-)$

(3) $\boldsymbol{P}_k = (I - \boldsymbol{K}_k\boldsymbol{H})\boldsymbol{P}_k^-$

Measurement update

Exercise 1

Consider a Kalman Filter in 1D domain. From kinematics you know

$$x_k = x_{k-1} + v_{k-1}\Delta t + a\Delta t^2/2$$
$$v_k = v_{k-1} + a\Delta t$$

Let's define $\boldsymbol{x}_k = \begin{bmatrix} x_k \\ v_k \end{bmatrix}$. Assume that the sensor aims at measuring directly the variable of concern.

Write the corresponding state update equation, i.e., give the matrices $\boldsymbol{A}$, $\boldsymbol{B}$ and $\boldsymbol{H}$.

Exercise 2

Still in reference to the previous exercise, apply the Kalman filter to track the object. Assume that

$$\hat{\boldsymbol{x}}_0^- = \begin{bmatrix} -1 \\ -2 \end{bmatrix} \quad \boldsymbol{P}_0 = \begin{bmatrix} 2 & 0 \\ 0 & 3 \end{bmatrix} \quad \boldsymbol{Q} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad \boldsymbol{R} = \begin{bmatrix} .5 & 0 \\ 0 & .5 \end{bmatrix}$$

$\boldsymbol{k}$	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
$\boldsymbol{z_k}$	1	4	9	16	25	36	49	64	81
	2	4	6	8	10	12	14	16	18

Write a simple MATLAB program that implements the Kalman, and computes $\hat{\boldsymbol{x}}_k^-$, $\boldsymbol{P}_k$, $\boldsymbol{K}_k$, $\hat{\boldsymbol{x}}_k$ and $\boldsymbol{P}_k^-$ for k=1,…, 9.

References

Useful link

- *Kalman Filter Learning Tool*. http://www.cs.unc.edu/~welch/kalman/

Books and Papers

- Kalman, R. E. 1960. “A New Approach to Linear Filtering and Prediction Problems,” Transaction of the ASME—Journal of Basic Engineering, pp. 35-45 (March 1960).
- Brown, R. G., Hwang, P.Y.C., Random Signals and Applied Kalman Filtering with MATLAB Exercises, 4th Ed., Johna Wile & Sons, Inc, 2012.

References

Online lectures

- Graphical Models 3 – Christopher Bishop.

MATLAB demos

Tracking with Kalman Filter – MATLAB demos (version 2015 onwards):

- kalmanFilterForTracking
- multiObjectTracking()

Kalman Filter

END